遺伝的アルゴリズムによるグラフ分割問題の解法

河本　敬子・片山　謙吾*・成久　洋之*

岡山理科大学大学院工学研究科修士課程情報工学専攻

*岡山理科大学工学部情報工学科

(1998年10月5日　受理)

1．はじめに

組合せ最適化問題の1つである，グラフ分割問題（Graph Partition Problem：GPP）はLSI設計（レイアウト等）において非常に重要な問題であるが，これを解くための計算量は膨大であり，実用的な規模の回路において最適解を求めることは困難である。遺伝的アルゴリズム（Genetic Algorithm：GA）は自然淘汰理論に基礎をおく生物の進化過程を模倣した工学的モデルであり，従来の手法では解決が困難であったさまざまな最適化・探索の問題に対して，許容される近似解を得ることができる有効な手法である。

本研究では，グラフ分割問題にGAを適用したときの有効性をみるために，既存のヒューリスティック手法として知られているシミュレーテッド・アニーリング法（Simulated Annealing：SA）と，MSLS（Multi Start Local Search）の各手法と比較検討するものである。

2．グラフ分割問題

2．1　グラフ分割問題について

グラフ分割問題とは，幾つかのノードとノード間を結ぶ何本かのリンクで構成されたグラフのノードを幾つかのグループに分割し（各グループ間に割り当てられるべきノード数は決められている），グループ間に跨るリンク数が最小となるような分割を求める問題である（図1）。

2．2　グラフ分割問題への定式化

グラフ分割問題は以下のように表現される。

$$G=(U,\ V),\quad U=\{n_j;\ j=1,\ 2,\ \ldots,\ N\},\quad V=\{e_{ij};\ n_i\in U,\ n_j\in U,\ i\neq j\}$$

ここで，U がノード集合，V が枝集合である。

グラフ G のノード集合を M 個の部分集合に分割する。

評価基準としては

(1)　部分集合間にまたがる枝数の総数を最小にする。

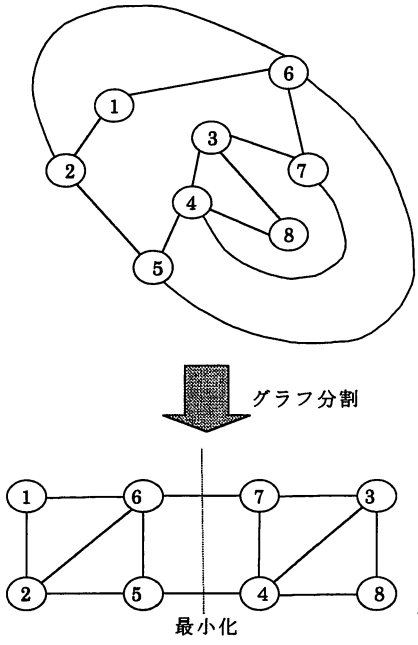

図1　グラフ分割問題

(2)　各部分集合のノード数をなるべく均一にする。

ノード i の状態を X_i で表す。即ち

$X_i = m$ は，ノード i が部分集合 m に割り当てられることを意味する。

基準(1)に対して関数

$$\phi_c(x_c) = \phi_c(x_i,x_j) = \begin{cases} 0 & if\ \ n_i = x_j \\ \omega_{ij} & otherwise \end{cases}$$

を導入する。

ここで，C^*は枝に対応するリンク（2-クリーク）の集合であり，ω_{ij} はノード n_i と n_j 間の枝の重みである．変数 n_i と n_j 間の枝がない場合は $\omega_{ij} = 0$ とする。

これにより基準(1)は次の式で表すことができる。

$$\sum_{c \in C} \phi_c(x_c) \rightarrow \min$$

基準(2)に対して次の関数を導入する。

$$\zeta(x,y) = \begin{cases} 0 & if\ \ x \neq y \\ 1 & otherwise \end{cases}$$

これにより基準(2)は次の式で表すことができる。

$$\sum_{m=1}^{M}\left[\sum_{i}\zeta(x_i,m)\right]^2\rightarrow\min$$

これを次のように変形する。

$$\begin{aligned}\sum_{m=1}^{M}\left[\sum_{i}\zeta(x_i,m)\right]^2&=\sum_{m}\left[\sum_{i}\zeta(x_i,m)^2+2\sum_{i<j}(x_i,m)\zeta(x_i,m)\right]\\&=\sum_{i}\sum_{m}\zeta(x_i,m)^2+2\sum_{i<j}\sum_{m}\zeta(x_i,m)\zeta(x_i,m)\\&=N+2\sum_{i<j}\zeta(x_i,x_j)\end{aligned}$$

従って GPP の目的関数は次の式となる。

$$f(x)=\sum_{c\in C}f_c(x_c)=\sum_{c\in C}\phi_c(x_c)+2\lambda\sum_{i<j}\zeta(x_i,x_j)$$

ここで λ は制約条件の強さを表す定数である。
明らかに，C には2-クリークしか含まれていない。

3．遺伝的アルゴリズム

3．1 遺伝的アルゴリズムについて

GA とは生物の進化のメカニズムにならった探索アルゴリズムである。このアルゴリズムは次のようなものである。まず問題を文字列 (string) に変換する。これは，いわば遺伝子型に対応するものである。そして文字列の集団 (population) を取り扱うのである。この文字列を評価して評価値の高い集団を選んで残すようにする。これは自然界における淘汰に対応する。この選ばれた集団に対してオペレータを施すことによって新しい文字列を生成する。基本的なオペレータは文字列を複製する機能を持つ自己生成(copy)，二つの文字列に対して部分的な交換を行うことによって新しい文字列を生み出す交叉(crossover)，文字列を複製するときに確率的に誤りを生じさせる突然変異 (mutation) 等がある。このサイクルを繰り返すことによって，環境に応じた評価値の高い文字列を生み出し，文字列の集団全体の評価値を向上させていくものである。

3．2 GA のアルゴリズム

ここでは，本研究で用いた GA のアルゴリズムについて述べる。

3．2．1 遺伝子型の決定

対象とする問題を遺伝子の形で表現する。本研究では，各ノード番号に対応した分割番号で個体をコード化する。その方法を以下に説明する。ノード数8を4つのサブグラフに分割した場合を考える。図2の分割の場合，図3のように，ノード3がサブグラフ1に，ノード1，6がサブグラフ0に，ノード5，7がサブグラフ2に，ノード0，2，

4がサブグラフ3に分割されていることを表す。そして，遺伝的操作を行うために図3の表現型を0と1のビット列の遺伝子型に変換する。

3.2.2 初期遺伝子集団の設定

本研究では，初期遺伝子集団をグラフのノード数に合わせて(60，40，20個)，ランダムに生成する。

3.2.3 適応度の評価

各個体の適応度の評価は，各個体の適応度をあらかじめ定めた適応度関数によって評価される。本研究の適応度は，目的関数の値を最小化し，適応度関数の値を最大化することから，以下の式で求められる。

目的関数

$$f(x) = g(x) + \lambda h(w)$$

λ ：重み

$g(x)$：部分集合に跨る枝数の総数の最小化を表す項

$h(x)$：各部分集合のノード数の均一化程度を表す項

適応度関数

$$f'(x) = 1/f(x)$$

3.2.4 選　　択

適応度の高い個体が次の世代により多くの子孫を残すという自然淘汰 (natural selec-

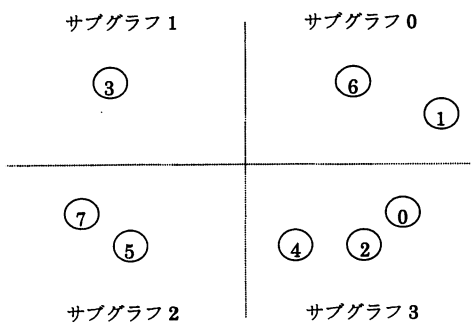

図2　個体の分割状態

ノード	0	1	2	3	4	5	6	7
個　体	3	0	3	1	3	2	0	2

図3　ランダムに生成された個体

tion）の考えはGAでは，選択（selection）あるいは再生（reproduction）ともよばれ，個体の中で問題（環境）への適応度の高いものは増殖し，逆に低いものは淘汰される。選択は，このような自然淘汰考えに対応する。

確率に従って個体を選択して交叉や突然変異を行う場合には，非常に良い個体が現れてもすぐに消滅してしまうことがある。このことは，確率的な操作をする以上やむを得ないことであり，また局所解に陥ることを避けることにもつながるが，現実に少ない回数で良い解を得たい場合には好ましくない。

また，確率的選択での問題点は，個体数が十分に多くないときには，乱数の揺らぎによって適応度を正確に反映しない選択がなされる可能性があるということである。

本研究では，これらの問題点の改善のためにエリート保存選択（elitist preserving selection）と期待値選択（expected-value selection）を2世代に1回用いる。期待値選択を行う際に，べき乗スケーリング（$f' = f^2$）を行う。

エリート保存選択は，個体群の中で最も適応度の高い個体は無条件でそのまま次世代に残すという選択あるいはそれに類似の選択方法である。

エリート保存選択については，集団中で最も適応度の高い個体を個体数の10%をそのまま次世代に残す方法を採用した。このエリート選択を，次のように定義する。

「$s^*(t)$ を，世代 t までに現れた最良の個体とする。もし，$X(t+1)$ を通常の方法で生成したときに，$X(t+1)$ の中に，$s^*(t)$ が存在しなければ，$s^*(t)$ を $X(t+1)$ の $N+i$ 番目の個体として加える。ここで，$i = 1,...,n$ とし，n は個体数の10%とする。」

エリート選択を採用すれば，その時点での最良の個体は，交叉や突然変異により破壊されることはないという利点がある。しかし，エリート個体の遺伝子が個体群の中に急速に広がる可能性が高いので局所的な解に陥る危険も含んでいる。

期待値選択は，個体群の中の各個体の適応度とその総計を計算し，適応度の総計に対する各個体の割合で個体数を調整するという基本的な考えに基づいている。

3.2.5 交　　叉

選択された個体間での染色体の組替えによる新しい個体を生成するという交叉(crossover）は，GAでは最も重要な役割を果たす遺伝的オペレータであるといえる。このような交叉は，個体群の中から任意の2つの個体（親）をランダムに選び，さらに，ランダムに選ばれた1点あるいは多点の交叉点（crossover point）で遺伝子を組替えることにより，新たな2つの個体（子）を生成する操作である。

したがって，交叉では，どのようにして任意の2つの個体を生成するのか，選択した2つの個体をどのように交叉させるのか，交叉により生成された新しい個体をいかにして個体群の中に組み込むのかという，3つの操作が重要となる。これらの操作は，それぞれ，交叉確率(probability of crossover) p_c, 交叉点の数 CP, 世代ギャップ(generation gap) G（入れ替える新個体の割合）に関係するが，このような交叉のアルゴリズムは，

以下に要約する。

本研究での交叉のアルゴリズム

手順1　$i=1$ として，i 番目の個体と交叉する個体を個の個体からなる個体群の中から選ぶ.

手順2　[0,1] の乱数 *rand* ( ) を発生させ，$Pc \geqq$ *rand* ( ) なら手順3へ，そうでなければ手順4へ行く。

手順3　2つの個体をある交叉方法で交叉させ，手順5へ行く。

手順4　交叉させなかった2つの個体を一時的に保存して，手順6へ行く。

手順5　交叉させた2つの新しい個体を一時的に保存して，手順6へ行く。

手順6　$i<N$ なら $i=i+1$ として手順1へ戻り，そうでなければ手順7へ行く。

手順7　保存されている $2\cdot N$ 個の個体の中から $N\cdot G$ 個 $(0<G\leqq 1)$ をランダムに選び，もとの N 個の個体群の中の $N\cdot G$ 個の個体と入れ替える。

このような交叉の規則は，個体の表現方法に依存していくつか提案されてきているが，本研究では，1点交叉（one-point crossover）または，単純交叉（simple crossover）とよばれる最も単純な交叉規則を用いる。1点交叉では，親1，親2の文字列上で交叉点（crossover point）“｜”をランダムに1箇所選び，交叉点の右側の2つの親の部分文字列をそっくりそのまま交換して，子1，子2を生成する。このような1点交叉の図4のようになる。

3.2.6. 突然変異

交叉だけでは，個体の親に依存するような限られた範囲の子しか生成することができない。突然変異（mutation）は，染色体上のある遺伝子座の値を他の対立遺伝子に置き換えることにより，交叉だけでは生成できない子を生成して，個体群の多様性を維持する働きをする。

2値 {0,1} の文字列，すなわちビット列で表される個体に対する GA の突然変異は，個体群の中の各個体の各遺伝子座の遺伝子に対して定められた突然変異確率 P_m で他の対立遺伝子に置き換えることによって行われる。例えば，文字列 101110100 で表される個体に対して第3遺伝子座に突然変異が起これば図5のようになる。ここで，本研究での突然変異のアルゴリズムを以下に要約する。

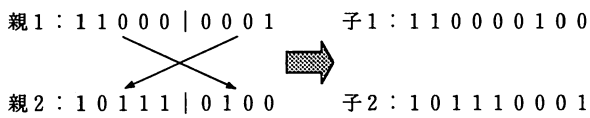

図4　交叉の例

図5　突然変異の例

突然変異のアルゴリズム

手順1　2値{0,1}の文字列，すなわちビット列で表される個体の各遺伝子座に対して［0,1］の乱数 *rand*（）を発生させる。

手順2　$P_m \geqq rand$（）ならば，遺伝子座の遺伝子を対立遺伝子に置き換える。すなわち0ならば1，1ならば0とする。

手順3　個体群の中のすべての個体に対して手順1，2を行って手順4へ行く。

手順4　適応度の高い，個体数の5％分の各個体の遺伝子座の遺伝子をランダムに選び，対立遺伝子に置き換えて終了する。

ここで，突然変異の概念はランダム探索の考えそのものに他ならないので，GA の特徴は突然変異よりはむしろ交叉にあることに注意する。また，突然変異は交叉により生成された優れた個体を破壊する可能性があるので，交叉確率 P_c と突然変異確率 P_m の関係については，$P_c > P_m$ に設定する必要がある。

ここで，GPP の場合，突然変異の操作は，ノードがある部分集合から，ある部分集合に移動することを意味する。

4．比較手法

4．1 シミュレーテッド・アニーリング法

この方法は統計力学において，溶解状態にある物質を冷却して結晶状態に到達させるプロセスからヒントを得たアルゴリズムである。このアルゴリズムでは，可能性のある探索点の中で最も有望な点を選んで探索を進めていく山登り法（Hill-Climbing：HC）に局所解（local minimum）へ陥るのを防ぐために，確率的な遷移を導入したものである。HC と違うところは，探索点 X_i の近傍を探索して解 X が得られたとき，評価値が X_i のそれよりも悪くても，$X_{i+1} = X$ とする可能性を残す点にある。すなわち，解が改善されれば必ずそれに置き換え，改善されない場合でも，確率 $\exp(^{-\delta C}/_{T})$ で置き換える。δC は評価値の改善値の改悪量，T は適当な定数で温度と呼ばれている。T が大きいほど，解が改善されて置き換えられる確率が大きくなる。収束を速めるために，最初は T を大きく設定しておき，少しずつ，を減らしてゆく。

4．2 Multi Start Local Search

ランダムな点から山登り法を用いて，一定の時間，最適解を目指して実行する方法である。

5. 実験内容

5.1 実験問題

本実験では，GA の有効性をみるために SA と，MSLS の各手法と比較検討した。適用する GPP は，試行回数 5 回毎に各アルゴリズムに対して，乱数の種を変えて実施する。

5.2 実験方法

グラフ分割問題は，ノード数を60，40，20に対して，枝数を180，120，60とし，各問題の分割数はすべて 4 とした。そして，それぞれのアルゴリズムについて 5 回の試行を行い，目的関数値（最小，平均，最大），計算時間を計算した。本実験では，Intel 社の MMX 搭載 Pentium 200MHz を使用して計算を実行した。

(1) GA

GA のパラメータは次のように設定する。

個体数　　：60，40，20

交叉確率 P_c　：1.0

世代ギャップ G　：0.5

突然変異確率 P_m ：0.3，0.1，0.05，0.01，0.005，0.004，0.003，0.0

打ち切り世代数は，400世代とする。

(2) SA

初期温度　50，最終温度　0.005，温度別計算数　600，温度低減率　0.98

(3) MSLS

予備実験により，GA，SA よりも長い計算時間として200秒の間局所探索法繰り返す。

5.3 実験結果

各表は，GPP の規模別の GA，SA，MSLS の実験結果である。各表から GA は，ノード数が60のとき P_m を0.004，ノード数40のときは P_m を0.005に，ノード数が20のときは P_m を0.01にしたとき上質な解を得ることができ，P_m の値が大きすぎたり小さすぎたりすると SA や MSLS よりも悪い解を得ることが分かる。

また，図 6 はノード数60，枝数180，分割数 4，pm0.005の GPP での GA と SA の比較である。各値は 5 回の試行の平均で，経過最小値の平均を表している。これより，GA は SA で算出された解よりも上質の解を短い時間で得ること示した。

図 7 は，P_m による目的関数値の変化を示したものである。突然変異を全く加えない P_m が0.0の場合，P_m が0.004よりも若い世代で良い解が得られているようにみえるが，初期収束を起こしている。そして，加えすぎた P_m が0.3の結果からは，加えすぎると良くないことが分かる。このことから，P_m の設定は，適用する GPP の規模によって最適な P_m が異なることを示した。

表1　実験結果（個体数・ノード数60，枝数180，分割数4）

計算法	Pm	最小値	平均値	最大値	計算時間
GA	0.3	2150.5	2151.68	2152.3	140秒
	0.1	2150.6	2151.54	2152.2	139秒
	0.05	2146.9	2149.5	2151.4	139秒
	0.01	2143.1	2144.96	2144.9	138秒
	0.005	2140.2	2141.76	2143.5	137秒
	0.004	2135.6	2138.18	2139.4	138秒
	0.003	2136.4	2139.28	2141.8	137秒
	0.0	2151.1	2153.8	2155.7	134秒
SA		2140.9	2142.88	2146.0	117秒
MSLS		2150.6	2152.26	2153.9	200秒

表2　実験結果（個体数・ノード数40，枝数120，分割数4）

計算法	Pm	最小値	平均値	最大値	計算時間
GA	0.3	931.0	932.06	933.2	77秒
	0.1	931.8	932.04	932.3	77秒
	0.05	930.9	931.52	932.1	76秒
	0.01	928.3	928.54	929.0	77秒
	0.005	921.9	923.56	925.1	76秒
	0.004	923.0	928.64	934.2	76秒
	0.003	927.4	930.36	933.2	75秒
	0.0	933.4	934.46	935.8	75秒
SA		927.8	928.10	928.8	71秒
MSLS		929.8	931.58	932.5	200秒

表3　実験結果（個体数・ノード数20，枝数60，分割数4）

計算法	Pm	最小値	平均値	最大値	計算時間
GA	0.3	215.4	216.34	217.0	47秒
	0.1	215.4	216.18	216.7	46秒
	0.05	214.5	215.58	217.1	46秒
	0.01	213.2	213.44	213.7	46秒
	0.005	216.2	217.22	218.2	46秒
	0.004	216.2	217.30	218.2	46秒
	0.003	215.3	216.72	217.7	46秒
	0	217.1	218.90	220.6	46秒
SA		214.2	214.82	215.5	50秒
MSLS		214.7	215.76	216.4	400秒

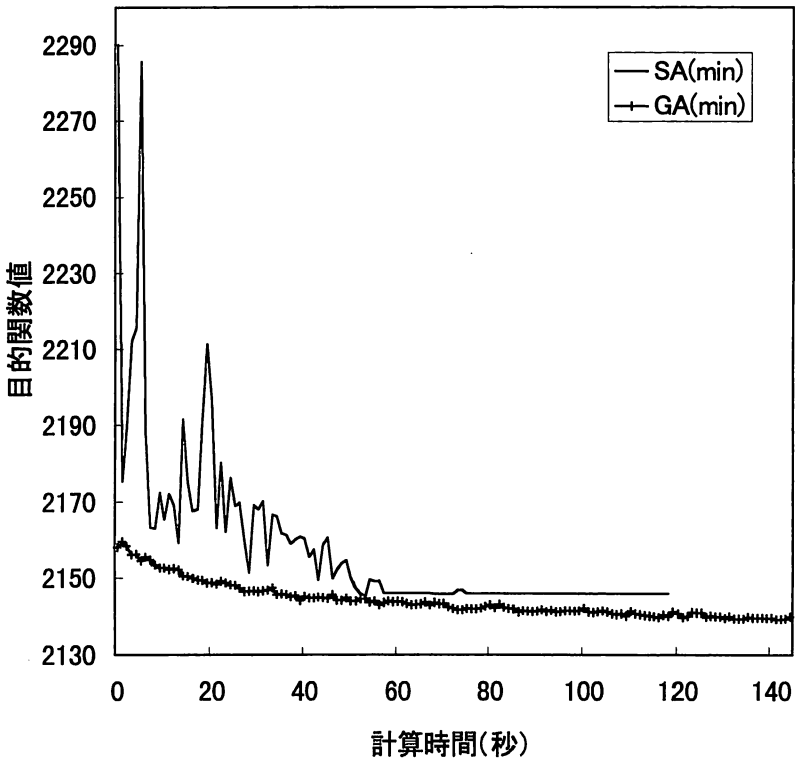

図6　実行時間比

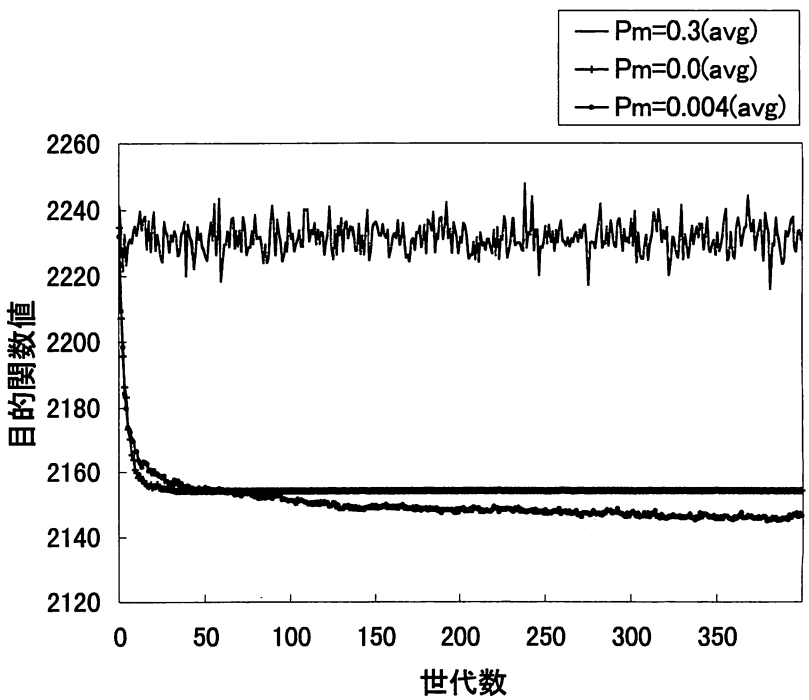

図7　Pm における目的関数値の変化

6. おわりに

本論文では,GPP に対する GA について記述した。比較アルゴリズムとして SA と MSLS を取り上げ，それらより GA は上質な解を算出可能であることを示した。今後の課題としては，GA の並列性について検討する予定である。

参考文献

1）坂和正敏：“遺伝的アルゴリズム”，朝倉書店（1996).
2）趙悦，何愛国：“組合せ最適化問題の分散シミュレーテッドアニーリング法”，広島電機大学研究報告，第29巻，1996.
3）北野宏明：“遺伝的アルゴリズム”，産業図書（1993).

An Algorithm for Solving the Graph Partition Problems by Genetic Algorithm

Keiko KOUMOTO, Kengo KATAYAMA* and Hiroyuki NARIHISA*
Graduate School of Engineering
**Department of Information & Computer Engineering*
Faculty of Engineering
Okayama University of Science
Ridai-cho 1-1, Okayama 700-0005, Japan
(Received October 5, 1998)

Graph Partition Problem (GPP), which is one of the combinational optimization problem, is very important in LSI design such as layout and so on. The volume of the calculation to investigate the whole solution of this problem is so large that it is difficult to solve optimization problem. Genetic algorithm (GA), which has been recently focussed on as the heuristic method, is one of the technological models imitated the process of evolution which based on the theory of natural selection. This is an efficient technique to get approximate solution for practical application, even though the conventional method was difficult to get the solution of various optimization and investigation problem.

In this paper, we investigate the efficiency of GA application for graph partition problems by comparing with two combinational heuristic algorithm; e.g. Simulated Annealing (SA) and Multi Start Local Search (MSLS) methods.